# Tentamen Concurrency, 22 april 2009

Tijdsduur 3 uur. Gesloten boek tentamen.

Voorzie alle in te leveren bladen van je naam, en nummer ze. Schrijf op het eerste blad het aantal ingeleverde bladen. Werk netjes, formuleer scherp en zorgvuldig. Schrijf duidelijk leesbaar.

Als het tentamen is nagekeken, kun je het komen inzien bij Wim H. Hesselink, Bernoulliborg kamer 374.

**Opgave 1** (30 %). Gegeven is een systeem met twee processen

```
       var  active[0: 1]: bool := ([2] false) ;
       var  turn: int := 0 ;

       process Partner (self := 0 to 1)
          do true ->
10:           NCS
11:           active[self] := true
12:           turn := 1 - self
13:           if active[1 - self] ->
14:               await turn = self
              fi
15:           CS
16:           active[self] := false
17:           turn := 1 - self
          od
       end Partner
```

`NCS` is de niet-critische sectie; dit commando hoeft niet te eindigen.
`CS` is de critische sectie, die altijd gegarandeerd eindigt en die alleen onder wederzijdse uitsluiting uitgevoerd mag worden.

(a) Geef een formele specificatie van wederzijdse uitsluiting voor dit systeem.

(b) Bewijs voor dit systeem invarianten van de vorm:

(J0) `active`$[q]$ $\equiv$ $q$ **in** $\{\ldots\}$ ,

(J1) $q$ **in** $\{\ldots\}$ $\wedge$ $r$ **at** 15 $\Rightarrow$ `turn` $=$ ? .

(c) Bewijs dat dit systeem aan wederzijdse uitsluiting voldoet, volgens je specificatie van onderdeel (a).

(d) Het algoritme wordt incorrect als de regels 11 en 12 worden verwisseld. Geef voor deze incorrecte versie een scenario waarin wederzijdse uitsluiting geschonden wordt.

Z.O.Z.

**Opgave 2** (30 %). Gevraagd wordt een systeem te ontwerpen met `nr` lezende threads en `nw` schrijvende threads volgens

```
thread Lezer(i:= 1 to nr)
  do true ->
10:     NCS
11:     lw.introR()
15:     CSR: lees data
16:     lw.exitR()
  od end
```

```
thread Schrijver(i:= 1 to nw)
  do true ->
20:     NCS
21:     lw.introW()
25:     CSW: schrijf data
26:     lw.exitW()
  od end
```

De lezers moeten tegelijk kunnen lezen, maar schrijvers mogen alleen schrijven als er geen andere lezers of schrijvers bezig zijn. Er moet dus gegarandeerd worden:

$$q \textbf{ at } 25 \ \wedge \ q' \textbf{ at } 25 \ \Rightarrow \ q = q' \ ,$$
$$q \textbf{ at } 25 \ \Rightarrow \ r \textbf{ not-at } 15 \ .$$

Als een schrijver aangeeft te willen schrijven, moeten nieuwe lezers tegengehouden worden. Als er geen schrijver schrijft of wil schrijven, moet elke nieuwe lezer toegelaten worden.

Implementeer hiervoor een Java class voor het object `lw` met de vier synchronisatiemethoden die in de regels 11, 16, 21 en 26 gebruikt worden. Je mag hierbij uitsluitend standaard Java-primitieven gebruiken (en daarin geen semaforen implementeren). Geef aan waarom je oplossing correct is.

**Opgave 3** (40 %). Gegeven zijn gehele getallen $1 \leq \mathtt{M} \leq \mathtt{N}$ en een systeem met `N` processen van de vorm:

**var** n : *int* := 0 , k : *int* := 0 ;
  b : *bool* := *false* ;

**process** *Member* (*self* := 0 **to** $N - 1$)
  **do** *true* $\rightarrow$
10:  *TNS0*
11:  ⟨ **await** ¬b **then** n ++ ; b := (n = $M$) ⟩
12:  ⟨ **await** b ⟩
13:  *TNS1*
14:  ⟨ k ++ ⟩
15:  ⟨ **await** k = n **then** k −− ; n −− ; b := (n ≠ 0) ⟩
  **od**
**end** *Member* .

De commando's *TNS0* en *TNS1* zijn gegeven. Ze wijzigen de gedeelde variabelen `k`, `n`, `b` niet en zijn verder niet van belang.
(a: 20%) Formuleer en bewijs invarianten die de waarden van `n`, `k`, `b` en $M$ relateren aan elkaar en aan het aantal processen die op bepaalde locaties in hun programma zijn. Deze invarianten moeten sterk genoeg zijn om daarmee te bewijzen dat geldt:

(*) $\qquad q \textbf{ at } 13 \ \Rightarrow \ \#\{r \mid r \textbf{ in } \{12 \ldots 15\}\} = M \ .$

Bewijs dat (*) inderdaad uit je invarianten volgt.
(b: 20%) Implementeer dit systeem met één of meer gesplitste binaire semaforen.